**TwinVerse - Project Kickoff & Development Plan**

**Project Type: Hackathon MVP (Walmart Sparkathon 2025)  
Tech Stack: React.js (frontend), Django + SQLite (backend), Firebase (chat), Axios (API), JWT (Auth), Figma (UI/UX)**

**✅ Project Goals**

* **Build a 2D digital twin shopping social platform**
* **Let users create digital twins through onboarding quiz**
* **Generate daily personalized product recs**
* **Enable social feed, friend network, groups, leaderboard, chat**
* **Offer gamification, incognito purchase, and influencer-style twins**

**👥 Team Division**

* **Team 1 (Frontend): React, Axios, Tailwind, Firebase, UI from Figma**
* **Team 2 (Backend): Django REST, SQLite DB, JWT auth, APIs**

**🔁 Parallel Development Strategy**

* **Define shared API contracts early (API endpoints, data shape)**
* **Frontend uses mocked/dummy JSONs initially, switches to real APIs once ready**
* **Use Postman collection to test endpoints independently**
* **Daily sync to resolve API/field mismatches**

**🛠️ 0. Initial Setup (Day 0)**

**🔹 Team 1: Frontend Setup**

1. **Create React project:**

**npx create-react-app twinverse-client**

**cd twinverse-client**

**npm install axios react-router-dom firebase tailwindcss**

**npx tailwindcss init**

1. **Tailwind Setup:**

* **Edit tailwind.config.js**
* **Include in index.css**

1. **Set up routing + pages:**

* **/onboarding**
* **/profile**
* **/feed**
* **/groups**
* **/leaderboard**

1. **Firebase setup for chat**

**🔹 Team 2: Backend Setup**

1. **Create Django project:**

**pip install django djangorestframework djangorestframework-simplejwt**

1. **Create project + apps:**

**django-admin startproject twinverse\_server**

**cd twinverse\_server**

**python manage.py startapp users**

**python manage.py startapp twins**

**python manage.py startapp feed**

1. **Add to INSTALLED\_APPS, migrate, run server**
2. **Set up /api/token/ auth endpoints**

**🧭 Phase-Wise Plan (Parallel Execution)**

**🚀 Phase 1: Auth System**

**Team 2 (Backend):**

* **JWT login/register APIs**
* **Setup custom user model**

**Team 1 (Frontend):**

* **AuthContext to manage tokens**
* **Forms for login/register**
* **Store token in localStorage**

**🎭 Phase 2: Twin Creation**

**Team 2:**

* **TwinProfile model**
* **/api/twin/profile/ POST & GET**

**Team 1:**

* **Onboarding quiz UI**
* **Twin card visual**
* **Use dummy JSON if API not ready**

**🎁 Phase 3: Recommendations**

**Team 2:**

* **Logic for daily recs based on profile**
* **/api/twin/recommendations/**

**Team 1:**

* **Show carousel of 3–5 daily picks**
* **Add reason: “Your twin saw 10 others buy this”**

**🌐 Phase 4: Social Feed**

**Team 2:**

* **PurchaseLog model**
* **Public feed API**
* **Private purchase flag**

**Team 1:**

* **Feed UI with card scroll**
* **Like/react buttons**
* **Dummy data fallback**

**🤝 Phase 5: Friends & Groups**

**Team 2:**

* **Friendship model**
* **Groups model + API**
* **Group purchase voting**

**Team 1:**

* **Friends tab + group tab**
* **UI for adding people**
* **Group cart mock vote logic**

**📊 Phase 6: Leaderboard & Celeb Twins**

**Team 2:**

* **Compute leaderboard API**
* **Static celeb twin data**

**Team 1:**

* **Rank UI with badges**
* **Follow celeb twins**

**📱 Phase 7: Final Polish + Firebase**

**Team 1:**

* **Firebase SDK setup**
* **Use push() and onValue()**
* **Emojis, QR sharing, twin stories**

**Team 2:**

* **API cleanup & optimization**
* **Add default dummy data**

**🔗 Integration Strategy**

1. **Shared API Spec Document**
   * **JSON structure of all endpoints**
   * **Update whenever API changes**
2. **Mock API JSON**
   * **Team 2 provides sample responses**
   * **Team 1 uses them as fallback until live**
3. **Use Postman for Testing**
   * **Test all endpoints separately**
   * **Team 1 verifies data flow**
4. **Folder Syncing**
   * **Maintain frontend/ and backend/ directories**
   * **Use Git branches for separate features**
5. **Backend Deployment (Optional for frontend testing)**
   * **Use ngrok or localhost tunnel so frontend team can test with backend live**

**🔌 Firebase Chat Integration**

**(Solely handled by Team 1)**

* **Firebase Console → Project**
* **Realtime DB setup**
* **Firebase SDK + chat UI**
* **Chat data linked to friend ID or group ID**

**🎨 Using Figma in React (Team 1)**

* **Figma mockups for all screens**
* **Use Tailwind to match spacing/colors**
* **Export icons/images for /assets/**

**📡 Axios Setup (Team 1)**

**// api.js**

**const API = axios.create({ baseURL: 'http://localhost:8000/api/' });**

**API.interceptors.request.use(config => {**

**const token = localStorage.getItem('access\_token');**

**if (token) config.headers.Authorization = `Bearer ${token}`;**

**return config;**

**});**

**export default API;**

**🧪 Testing Checklist**

* **✅ Token saved, user persisted**
* **✅ Feed works with/without incognito**
* **✅ Twin shown after onboarding**
* **✅ Firebase sends/receives chat in real-time**

**🏁 Deployment (if needed)**

* **Frontend → Vercel (React)**
* **Backend → Render (SQLite okay)**
* **Firebase → Auto-hosted**

**Prepared by: Amaresh & ChatGPT  
Last Updated: June 18, 2025**